Программирование

Python

Создание приложений с графическим интерфейсом пользователя. TKinter. Qt.

Черновик

Кафедра ИВТ и ПМ
ЗабГУ

2018

План

GUI

Графический интерфейс пользователя, графический пользовательский интерфейс (ГПИ) (**graphical user interface, GUI**) — разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

Tkinter (**Tk inter**face) — кросс-платформенная графическая библиотека на основе библиотеки Tk (широко распространённая в мире GNU/Linux и других UNIX-подобных систем, портирована также и на Microsoft Windows).

Tk

Outline

Событийно-ориентированное программирование

Приложения с графическим интерфейсом как правило строятся с использованием событийно-ориентированного программирования (event-driven programming).

Событийно-ориентированное программирование - парадигма программирования, в которой программы определяется событиями — действиями пользователя (клавиатура, мышь), сообщениями других программ и потоков, событиями операционной системы (например, поступлением сетевого пакета)

Событийно-ориентированное программирование

- Порядок выполнения программы в известной степени определяет не программист, а пользователь.
- Воздействуя на элементы интерфейса пользователь генерирует ***события*** в программе.
- К каждому элементу интерфейса, на который пользователь должен воздействовать, привязывается ***обработчик события***.
- Обработчик события - это функция, которая вызывается в ответ на данное событие.

Событийно-ориентированное программирование

- Программа с GUI должна следить за действиями пользователя и вызывать соответствующие обработчики.
- Часть программы, решающая эту задачу, на называется главным циклом обработки событий (event loop, main loop в tkinter).
- Такой цикл должен быть в любой программе с GUI, поэтому его включают в фреимворки языков программирования и разработчикам приложений не приходится создавать его самостоятельно.

Outline

Структура программы с GUI

Пример 1

Пустое окно на Tkinter

```
from tkinter import *

# создать окно
root = Tk()

# создание отдельных элементов интерфейса пользователя
# ...

# запуск главного цикла обработки событий
root.mainloop()

# после вызова метода mainloop дальнейшие команды python
# исполняться не будут
```

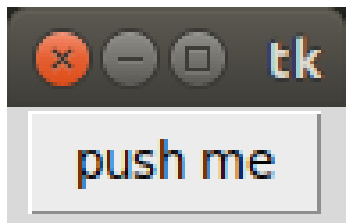
Окно с кнопкой

Пример 2

```
from tkinter import *

# создать окно
root = Tk()

# создать кнопку (пока не видна)
# 17, 5 - ширина и высота в знакоместах
button = Button(root, width=17, height=5,
                                    text = "push me!")

# расположить кнопку на окне
button.pack()

root.mainloop()
```

Окно с кнопкой

Пример 2



Создание обработчиков событий

К нажатию на кнопку можно привязать функцию, которая будет вызвана в соответствующий нажатию момент времени.

- Нажатие на кнопку - **событие**
- Функция, которая вызывается в ответ на возникновение события - **обработчик события**

Для привязки обработчика к событию используется метод bind:

```
имя_компонента.bind( событие, обработчик )
```

Подробнее http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm

Создание обработчиков событий

Для привязки обработчика к событию используется метод bind:

```
имя_компонента.bind( событие, обработчик )
```

Привязка события *Нажатие на левую кнопку мыши (Button-1)* к обработчику button_clicked

```
def button_clicked(event):
    pass

# ...
button = Button(root, text = "push me!")

button.bind('<Button-1>', button_clicked)
```

Некоторые события

Мышь и клавиатура

- <Button-1> клик левой кнопкой мыши (2 - средняя кнопка, 3 - правая кнопка)
- <ButtonRelease-1> - левая кнопка отпущена
- <Double-Button-1> - двойной клик левой кнопкой мыши
- <Enter> - указатель мыши наведён на виджет
- <Leave> - указатель мыши убран с виджета

- <Return> - нажат Enter
- <Key> - нажата клавиша Key (например <x> нажата клавиша x, <F5> - клавиша F5)
- <Control-key> нажата клавиша Key и клавиша Control (например <Ctrl-c> нажато сочетание клавиш Ctrl+c)
- <Alt-key> нажата клавиша Key и клавиша Alt (например <Ctrl-c> нажато сочетание клавиш Alt+c)

Некоторые события

другие события:
https://stackoverflow.com/questions/32289175/list-of-all-tkinter-events

Окно с кнопкой

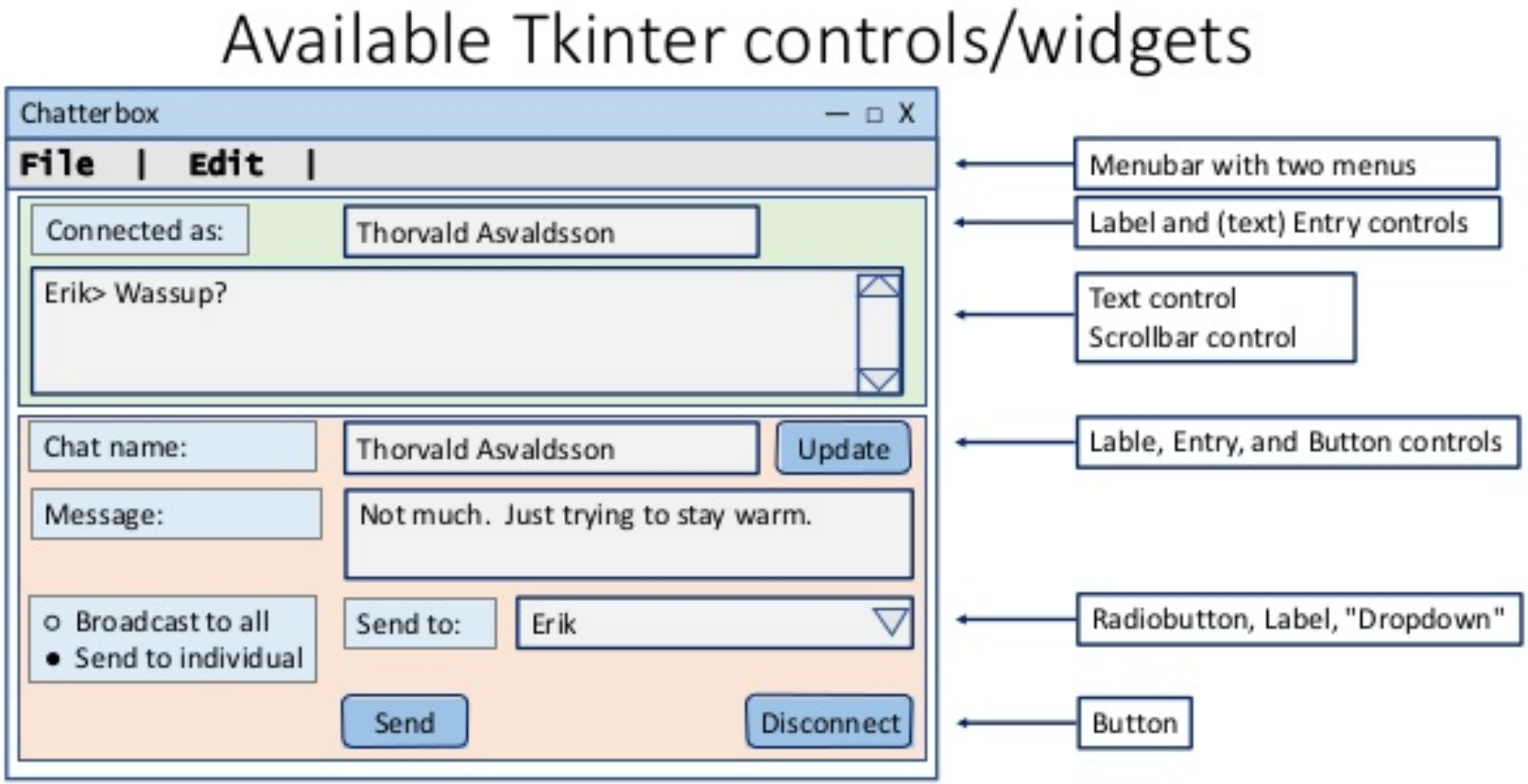
Пример 2.1

Добавим обработчик нажатия кнопки. После нажатия на кнопку на ней будет меняться надпись: появится случайное число

```
from tkinter import *
from random import randint

# обработчик нажатия на кнопку
# имя функции выбирается произвольно
def button_clicked(event):
        global button
        button['text'] = str( randint(0,99) )

root = Tk()
button = Button(root, width=17, height=5,
        text = "push me!")
# расположить кнопку на окне
button.pack()
# связать событие для кнопки (клик левой клавишей мыши -
# <Button-1>) и обработчик события button_clicked
button.bind('<Button-1>', button_clicked)
root.mainloop()
```

Виджеты

В фреймворке Tkinter элементы интерфейса называются **виджетами**.



For more info, see https://docs.python.org/3/library/tk.html and http://tkinter.unpythonic.net/wiki/ and https://docs.python.org/3/library/tkinter.ttk.html

Основные виджеты

- Button - кнопка
- Label - лейбл (надпись)
- Entry - поле ввода
- Text - многострочное текстовое поле
- Listbox - список с множественным выбором
- Frame - пустой виджет
- Checkbutton - "чекбокс"(флажок)
- Radiobutton - переключатель
- Scale - шкала
- Scrollbar - полоса прокрутки

Поле ввода, надписи и одна кнопка

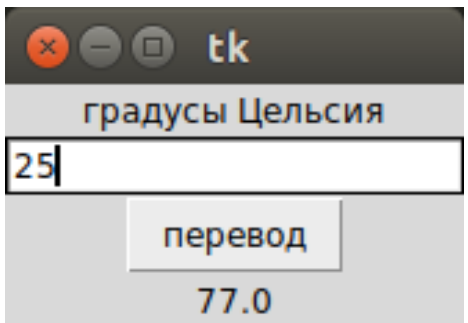
```
from tkinter import *

def calc(event):
        global edit, label_result
        c = float( edit.get() )
        f = 9/5 * c + 32
        label_result['text'] = str(f)

root = Tk()
label = Label(root, text = 'градусы Цельсия')
label.pack()
edit = Entry(root)
edit.insert(0,"25")
edit.pack()
button = Button(root, text='перевод')
button.pack()
label = Label(root, text = 'градусы Фаренгейта')
label_result = Label(root)
label_result.pack()

button.bind('<Button-1>', calc)
root.mainloop()
```

Поле ввода, надписи и одна кнопка

Как будет выглядеть окно программы с предыдущего слайда?

Поле ввода, надписи и одна кнопка

Как будет выглядеть окно программы с предыдущего слайда?



Картинки

Label можно использовать для отображения картинок

```
from tkinter import *
from PIL import Image, ImageTk

window = Tk()

image = Image.open("image-small.jpg")
img = ImageTk.PhotoImage(image)

label = Label(window, image = img)
label.pack()

window.mainloop()
```

Свойства виджетов

Указывать свойства виджета можно при создании, задавая значение параметра в конструкторе

```
l1 = Label(root, text="...", font="Arial 12")
```

Если виджет уже создан, то можно изменить его свойства использую оператор [], который работает так же как и для словарей.

```
l1['text'] = 'do you read me?'
```

Некоторые свойства виджетов

Многие виджеты имеют схожие свойства.

Перечислим некоторые из них

- Шрифт bold - полужирный
  italic - курсив
  дополнительные свойства шрифта (bold, italic) можно комбинировать или не указывать.
- размер[1]
  ширина высота

[1] размер определяется не в пикселях, а в занкоместах - размерах прямоугольника в который можно вписать один символ фиксированного размера

Упаковщик

Упаковщик (менеджер геометрии, менеджер расположения) это специальный механизм, который размещает (упаковывает) виджеты на окне.

В Tkinter есть три упаковщика: pack, place, grid.

Обратите внимание, что в одном виджете можно использовать только один тип упаковки, при смешивании разных типов упаковки программа, скорее всего, не будет работать.

Другие фреймворки

Tkinter предоставляет только базовые возможности для создания приложений с GUI.

Для сложных интерфейсов имеет смысл рассмотреть другие фреймворки:

-  wxPython
-  PyQt
- 

Qt versus Wx: How do two of the most popular Python frameworks compare?

Outline

Создание приложения на Python с использованием Qt

- Создание дизайна окна с помощью QtCreator
- Преобразование получившегося ui файла в код на Python
  `pyuic5 path/to/design.ui -o output/path/to/design.py`
- Создание класса на основе созданного класса окна предыдущего окна

Qt

Пример. Основная программа.

```
import sys
from PyQt5 import QtWidgets
from form import Ui_Form  # модуль полученный из ui файла


class Form(QtWidgets.QMainWindow, Ui_Form):
    pass


def main():
    app = QtWidgets.QApplication(sys.argv)
    form = Form()
    form.show()
    app.exec_();


if __name__ == '__main__':
    main()
```

Подробнее: https://tproger.ru/translations/python-gui-pyqt/

Outline

Ссылки и литература

- Tkinter - wikiversity
- younglinux.info/tkinter.php - Tkinter. Программирование графического интерфейса

Ссылки и литература

Ссылка на слайды

github.com/VetrovSV/Programming